Existe dois métodos que você pode usar para fazer o **INSERT** (Inserção) de dados na sua *database* MySQL. São eles: método PHP MySQLi e PHP Data Object (ou método PDO).

Método MySQLi

Primeiro de tudo, você deve estabelecer uma conexão com o banco de dados. Depois dessa etapa, podemos prosseguir com o **INSERT** do *query* do MySQL. Aqui está um exemplo de código completo com os métodos básicos de conexão e inserção:

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

```
<?php

$servername = "mysql.hostinger.co.uk";

$database = "u266072517_name";

$username = "u266072517_user";

$password = "buystuffpwd";

// Create connection

$conn = mysqli_connect($servername, $username, $password, $database);

// Check connection

if (!$conn) {

die("Connection failed: " . mysqli_connect_error());

}

echo "Connected successfully";

$sql = "INSERT INTO Students (name, lastname, email) VALUES ('Test', 'Testing',
'Testing@tesing.com')";

if (mysqli_query($conn, $sql)) {

echo "New record created successfully";

} else {
```

echo "Error: " . $sql . "
" . mysqli_error($conn);

}

mysqli_close($conn);

?>

<?php $servername = "mysql.hostinger.co.uk"; $database = "u266072517_name"; $username = "u266072517_user"; $password = "buystuffpwd"; // Create connection $conn = mysqli_connect($servername, $username, $password, $database); // Check connection if (!$conn) { die("Connection failed: " . mysqli_connect_error()); } echo "Connected successfully"; $sql = "INSERT INTO Students (name, lastname, email) VALUES ('Test', 'Testing', 'Testing@tesing.com')"; if (mysqli_query($conn, $sql)) { echo "New record created successfully"; } else { echo "Error: " . $sql . "
" . mysqli_error($conn); } mysqli_close($conn); ?>

```
<?php
$servername = "mysql.hostinger.co.uk";
$database = "u266072517_name";
$username = "u266072517_user";
$password = "buystuffpwd";
// Create connection
$conn = mysqli_connect($servername, $username, $password, $database);
// Check connection
if (!$conn) {
      die("Connection failed: " . mysqli_connect_error());
}

echo "Connected successfully";

$sql = "INSERT INTO Students (name, lastname, email) VALUES ('Test',
'Testing', 'Testing@tesing.com')";
if (mysqli_query($conn, $sql)) {
      echo "New record created successfully";
} else {
      echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}
mysqli_close($conn);
?>
```


A primeira parte do código (linhas **3 – 18**) é sobre a atual conexão do banco de dados. Não vamos passar pelo processo todo novamente. Contudo, se você deseja saber o que cada linha de código significa, confira nosso tutorial sobre **como se conectar a um banco de dados**.

Vamos começar pela linha **19**:

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

$sql = "INSERT INTO Students (name, lastname, email) VALUES ('Test', 'Testing', 'Testing@tesing.com')";

$sql = "INSERT INTO Students (name, lastname, email) VALUES ('Test', 'Testing', 'Testing@tesing.com')";

```
$sql = "INSERT INTO Students (name, lastname, email) VALUES ('Test', 'Testing', 'Testing@tesing.com')";
```

Esta é a linha mais importante do nosso código e que executa tudo o que a gente quer aprender neste tutorial – inserir dados MySQL PHP. O **INSERT INTO** é uma instrução que adiciona dados em banco de dados específicos da tabela. Neste exemplo, estamos adicionando dados à tabela **Students**.

Seguindo adiante, entre os colchetes, temos as colunas de tabela para as quais queremos adicionar valores (**name**, **lastname**, **email**). Os dados vão ser preenchidos na ordem especificada. Se escrevêssemos (email, lastname, name), os valores seriam adicionados na ordem errada.

A próxima parte é sobre a declaração **VALUES**. Aqui especificamos os valores para as colunas determinadas anteriormente. Dessa maneira, cada coluna representa um valor específico. Por exemplo, no nosso caso ele seria: **name = Thom, lastname = Vial, email = thom.v@some.com**.

Outra coisa relevante é que acabamos de executar a **SQL Query** usando um código PHP. Lembrando que as SQL Queries devem ser posicionados entre as citações. No nosso exemplo, tudo o que está entre citações e depois de $sql = é uma SQL Query.

A próxima parte do código (linhas **20 – 22**) mostra se nosso pedido foi feito com sucesso.

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

if (mysqli_query($conn, $sql)) {

echo "New record created successfully";

}

if (mysqli_query($conn, $sql)) { echo "New record created successfully"; }

```
if (mysqli_query($conn, $sql)) {
     echo "New record created successfully";
}
```

Ele simplesmente mostra uma mensagem de sucesso se o pedido que executamos foi realizado.

E a parte final (linhas **22 – 24**) mostra uma mensagem diferente caso o pedido não seja executado com sucesso:

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

else {

echo "Error: " . $sql . "
" . mysqli_error($conn);

}

else { echo "Error: " . $sql . "
" . mysqli_error($conn); }

```
else {
     echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}
```

Ele vai mostrar uma mensagem de erro caso algo dê errado.

Método PHP Data Object (PDO)

Assim como no exemplo anterior, precisamos primeiro de uma conexão com o banco de dados criando um novo objeto PDO. Para isso, use este tutorial.

Como a conexão ao banco de dados MySQL é um objeto PDO, você deve usar vários **métodos** PDO (qualquer função que seja parte de qualquer objeto) para preparar e fazer os pedidos. Os métodos de objetos são chamados assim:

```
$the_Object->the_Method();
```

O PDO permite que você **prepare** o código SQL antes dele ser executado. A SQL Query é avaliada e **corrigida** antes de ser executada. Um ataque de injeção de SQL simplificado poderia ser feito apenas escrevendo o código SQL em um campo de um formulário. Por exemplo:

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

// User writes this in the username field of a login form

john"; DROP DATABASE user_table;

// The final query becomes this

"SELECT * FROM user_table WHERE username = john"; DROP DATABASE user_table;

// User writes this in the username field of a login form john"; DROP DATABASE user_table; // The final query becomes this "SELECT * FROM user_table WHERE username = john"; DROP DATABASE user_table;

```
// User writes this in the username field of a login form
john"; DROP DATABASE user_table;
// The final query becomes this
"SELECT * FROM user_table WHERE username = john"; DROP DATABASE
user_table;
```

Como existe um código de SQL sintaticamente correto, o ponto e vírgula faz do **DROP DATABASE user_table** uma nova SQL Query, e a sua tabela é apagada. Declarações preparadas não permitem os caracteres de aspas (“) e nem ponto e vírgula (;) no final da solicitação original e a instrução maliciosa **DROP DATABASE** nunca será executada.

Você **sempre** deve usar declarações preparadas quando enviar ou receber dados do banco de dados com PDO.

Para usar declarações preparadas, você deve escrever uma nova variante que chame o método **prepare()** do objeto do banco de dados.

No código correto:

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

<?php

$servername = "mysql.hostinger.com";

$database = "u266072517_name";

$username = "u266072517_user";

$password = "buystuffpwd";

$sql = "mysql:host=$servername;dbname=$database;";

$dsn_Options = [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION];

// Create a new connection to the MySQL database using PDO, $my_Db_Connection is an object

```
try {
$my_Db_Connection = new PDO($sql, $username, $password, $dsn_Options);
echo "Connected successfully";
} catch (PDOException $error) {
echo 'Connection error: ' . $error->getMessage();
}
// Set the variables for the person we want to add to the database
$first_Name = "Thom";
$last_Name = "Vial";
$email = "thom.v@some.com";
// Here we create a variable that calls the prepare() method of the database object
// The SQL query you want to run is entered as the parameter, and placeholders are written like this :placeholder_name
$my_Insert_Statement = $my_Db_Connection->prepare("INSERT INTO Students (name, lastname, email) VALUES (:first_name, :last_name, :email)");
// Now we tell the script which variable each placeholder actually refers to using the bindParam() method
// First parameter is the placeholder in the statement above - the second parameter is a variable that it should refer to
$my_Insert_Statement->bindParam(':first_name', $first_Name);
$my_Insert_Statement->bindParam(':last_name', $last_Name);
$my_Insert_Statement->bindParam(':email', $email);
// Execute the query using the data we just defined
// The execute() method returns TRUE if it is successful and FALSE if it is not, allowing you to write your own messages here
if ($my_Insert_Statement->execute()) {
echo "New record created successfully";
} else {
echo "Unable to create record";
}
```

// At this point you can change the data of the variables and execute again to add more data to the database

$first_Name = "John";

$last_Name = "Smith";

$email = "john.smith@email.com";

$my_Insert_Statement->execute();

// Execute again now that the variables have changed

if ($my_Insert_Statement->execute()) {

echo "New record created successfully";

} else {

echo "Unable to create record";

}

<?php $servername = "mysql.hostinger.com"; $database = "u266072517_name"; $username = "u266072517_user"; $password = "buystuffpwd"; $sql = "mysql:host=$servername;dbname=$database;"; $dsn_Options = [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]; // Create a new connection to the MySQL database using PDO, $my_Db_Connection is an object try { $my_Db_Connection = new PDO($sql, $username, $password, $dsn_Options); echo "Connected successfully"; } catch (PDOException $error) { echo 'Connection error: ' . $error->getMessage(); } // Set the variables for the person we want to add to the database $first_Name = "Thom"; $last_Name = "Vial"; $email = "thom.v@some.com"; // Here we create a variable that calls the prepare() method of the database object // The SQL query you want to run is entered as the parameter, and placeholders are written like this :placeholder_name $my_Insert_Statement = $my_Db_Connection->prepare("INSERT INTO Students (name, lastname, email) VALUES (:first_name, :last_name, :email)"); // Now we tell the script which variable each placeholder actually refers to using the bindParam() method // First parameter is the placeholder in the statement above - the second parameter is a variable that it should refer to $my_Insert_Statement->bindParam(':first_name', $first_Name); $my_Insert_Statement->bindParam(':last_name', $last_Name); $my_Insert_Statement->bindParam(':email', $email); // Execute the query using the data we just defined // The execute() method returns TRUE if it is successful and FALSE if it is not, allowing you to write your own messages here if ($my_Insert_Statement->execute()) { echo "New record created successfully"; } else { echo "Unable to create record"; } // At this point you can change the data of the variables and execute again to add more data to the database $first_Name = "John"; $last_Name = "Smith"; $email =

"john.smith@email.com"; $my_Insert_Statement->execute(); // Execute again now that the variables have changed if ($my_Insert_Statement->execute()) { echo "New record created successfully"; } else { echo "Unable to create record"; }

```
<?php
$servername = "mysql.hostinger.com";
$database = "u266072517_name";
$username = "u266072517_user";
$password = "buystuffpwd";
$sql = "mysql:host=$servername;dbname=$database;";
$dsn_Options = [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION];
// Create a new connection to the MySQL database using PDO,
$my_Db_Connection is an object
try {
  $my_Db_Connection = new PDO($sql, $username, $password,
$dsn_Options);
  echo "Connected successfully";
} catch (PDOException $error) {
  echo 'Connection error: ' . $error->getMessage();
}
// Set the variables for the person we want to add to the database
$first_Name = "Thom";
$last_Name = "Vial";
$email = "thom.v@some.com";
// Here we create a variable that calls the prepare() method of the
database object
// The SQL query you want to run is entered as the parameter, and
placeholders are written like this :placeholder_name
$my_Insert_Statement = $my_Db_Connection->prepare("INSERT INTO
students (name, lastname, email) VALUES (:first_name, :last_name,
:email)");
// Now we tell the script which variable each placeholder actually
refers to using the bindParam() method
// First parameter is the placeholder in the statement above - the
second parameter is a variable that it should refer to
$my_Insert_Statement->bindParam(':first_name', $first_Name);
$my_Insert_Statement->bindParam(':last_name', $last_Name);
$my_Insert_Statement->bindParam(':email', $email);
// Execute the query using the data we just defined
// The execute() method returns TRUE if it is successful and FALSE if
it is not, allowing you to write your own messages here
if ($my_Insert_Statement->execute()) {
  echo "New record created successfully";
} else {
  echo "Unable to create record";
}
// At this point you can change the data of the variables and execute
again to add more data to the database
$first_Name = "John";
$last_Name = "Smith";
$email = "john.smith@email.com";
$my_Insert_Statement->execute();
// Execute again now that the variables have changed
if ($my_Insert_Statement->execute()) {
  echo "New record created successfully";
} else {
  echo "Unable to create record";
}
```

Nas linhas **28**, **29** e **30**, nós usamos o método **bindParam()** do objeto do banco de dados. Também existe um método **bindValue()** que é bem diferente.

- **bindParam() –** Esse método avalia dados quando o método **execute()** é alcançado. A primeira vez que o script alcança um método **execute()** ele vê que o **$first_Name** corresponde ao "Thom", vincula aquele valor e executa a consulta. Quando o script alcança o segundo método execute(), ele vê o **$first_Name** agora corresponde a "John", vincula aquele valor e executa a consulta novamente com os novos valores. O que é importante de se lembrar é que nós definimos a consulta uma vez a reusamos com dados diferentes em diferentes pontos do script.
- **bindValue() –** Esse método avalia os dados assim que o **bindValue()** é alcançado. Como o valor do **$first_Name** foi definido como "Thom" quando o **bindValue()** foi alcançado, ele será usado toda vez que um método **execute()** for chamado para **$my_Insert_Statement**.

Perceba que nós usamos a variante **$first_Name** e demos a ela um outro valor na segunda vez. Se você conferir o banco de dados depois de executar o script, você terá ambos os nomes definidos, apesar da variável **$first_Name** ser equivalente a "John" ao final do script. Lembre-se que o PHP avalia o script inteiro antes de realmente executá-lo.

Se você atualizar o script e substituir **bindParam** com **bindValue**, você vai inserir "Thom Vail" duas vezes no banco de dados e "John Smith" será ignorado.